COMP 523: Language-based security
Assignment 2 (100 points total)

Prof. B. Pientka
McGill University

September 15, 2010—**Due: Wednesday, 22 September 2010 at 2:35pm**

Exercise 1 (45pts): Extend the language for booleans and arithmetic expressions we have seen in class (see also Ch 3, CH 8 in Pierce) with an expression `leq` $t\ t'$ which allows us to check whether `t` is less than or equal to `t'`.

10 points Define small-step evaluation rules for `leq` $t\ t'$.

13 points Prove that the rules are deterministic. Justify which cases are impossible and why.

2 points Define a typing rule for `leq` $t\ t'$.

20 points Prove that progress and type preservation holds for this extension.

Exercise 2 (55pts): In this question, we write some simple programs in Beluga.

10 points Extend the small-step evaluator in small−step.bel to handle the expressions `leq`-construct following your small-step rules from Exercise 1.

25 points Complete the big-step evaluator implemented by the function eval : term [ ] −> valOpt [ ] in big−step.bel for arithmetic expressions including `leq`-construct. Make sure to define your big-step evaluation rules for `leq` in such a way that they behave the same way as in the small-step semantics.

20 points Implement a type inference engine for this language. Your function infer should have the following type:

```
rec infer : term [ ] −> tpOpt [ ]
```

(Extra credit) (10 points) Continuations allow us to write more efficient functions for type inference and evaluation. Implement the type inference engine using continuations.